

CLAIMS

1 1. A method for managing access to resources, comprising:
2 accessing a list of resource signatures, each of the resource signatures being
3 classified to indicate an accessibility status of a corresponding resource;
4 generating a verification signature for a requested resource;
5 comparing the verification signature for the requested resource to the list of
6 resource signatures; and
7 determining the accessibility status of the requested resource in accordance with
8 the accessibility status by which the resource signature matching the verification
9 signature is classified.

1 2. A method according to Claim 1, wherein the resources include applications
2 or programs.

1 3. A method according to Claim 1, wherein the resource signature for each of
2 the respective resources includes a manipulation of structure-related data from the
3 resource.

1 4. A method according to Claim 1, wherein the resource signature for each of
2 the respective resources includes a hash of function names from each of the respective
3 resources.

1 5. A method according to Claim 1, wherein generating the verification
2 signature for the requested resource includes:

3 retrieving data that uniquely identifies the resource;
4 sorting the retrieved data;
5 linking the sorted data; and
6 executing a mathematical manipulation of the linked data.

1 6. A method according to Claim 5, wherein the resources include applications
2 or programs.

1 7. A method according to Claim 5, wherein the retrieved data includes an
2 import table.

1 8. A method according to Claim 5, wherein the retrieved data includes
2 function names from an import table.

1 9. A method according to Claim 5, wherein the retrieved data includes
2 dynamic link library (DLL) names from an import table.

1 10. A method according to Claim 1, wherein a same procedure is followed to
2 generate each of the resource signatures and to generate a verification signature.

1 11. A method according to Claim 1, wherein the accessibility status of the
2 resources includes one of permissible or impermissible.

1 12. A method for generating an application identifier, comprising:
2 receiving a command to generate an identifier for an application;
3 retrieving an import table from the application;
4 sorting information from the retrieved import table; and
5 performing a cryptographic function on the sorted information.

1 13. A method according to Claim 12, wherein the import table is retrieved from
2 an executable of the application.

1 14. A method according to Claim 12, wherein sorting information from the
2 retrieved import table includes sorting function names according to at least one
3 predetermined criterion.

1 15. A method according to Claim 12, wherein performing a cryptographic
2 function on the sorted information includes performing a one-way hash function.

1 16. A method of restricting particular applications, comprising:
2 receiving a list of application fingerprints corresponding respectively to restricted
3 applications;
4 receiving a request to execute an application;
5 generating a confirmation fingerprint for the requested application;
6 comparing the confirmation fingerprint to the list of application fingerprints; and
7 restricting the requested application if the confirmation fingerprint matches one of
8 the application fingerprints respectively corresponding to restricted applications.

1 17. A method according to Claim 16, wherein generating a confirmation
2 fingerprint for the requested application includes:

3 retrieving data from an executable of the requested application describing linkages
4 to other applications;

5 sorting the retrieved data;

6 organizing the sorted information in a predetermined manner; and

7 hashing the organized information.

1 18. A method according to Claim 17, wherein the retrieved data includes an
2 import table.

1 19. A method according to Claim 17, wherein the sorted information includes
2 function names.

1 20. A method according to Claim 16, wherein the restricted applications are not
2 licensed.

1 21. An apparatus, comprising:
2 a licensing manager component to provide identification for an application and to
3 further assign a classification of the application as being restricted or unrestricted; and
4 a developer component to code an operating system to include the identification in
5 correspondence with the classification.

1 22. An apparatus according to Claim 21, wherein the licensing manager
2 component is to generate identification for an application using an import table from the
3 application.

1 23. An apparatus according to Claim 21, wherein the licensing manager
2 component is to provide a restricted classification for an unlicensed application.

1 24. An apparatus according to Claim 23, wherein, for an unlicensed
2 application, the licensing manager component is to:
3 retrieve an import table from an executable of the application;
4 sort and string together function identifiers from the import table; and
5 execute a cryptographic algorithm on the function identifiers.

1 25. An apparatus according to Claim 24, wherein the cryptographic algorithm
2 is a hashing algorithm.

1 26. An apparatus, comprising:
2 an interface to receive a request for a running state of an application;
3 an application identifier to generate an identifier for the application;
4 an application manager to match the identifier against a list of identifiers
5 indicating whether corresponding applications are eligible or ineligible for a running
6 state; and
7 an enabler to enable the running state for the application if the identifier is not
8 matched to an identifier indicating that the application is ineligible.

1 27. An apparatus according to Claim 26, wherein the application identifier is to
2 generate an identifier using an import table from the application.

1 28. An apparatus according to Claim 27, wherein the application identifier is
2 to:
3 retrieve an import table from an executable of the application;
4 sort and string together the function identifiers from the import table; and
5 hash the function identifiers.

1 29. An apparatus according to Claim 28, wherein the instruction to hash further
2 includes an instruction to execute an MD5 hashing algorithm.

1 30. A computer-accessible medium having one or more instructions that are
2 executable by one or more processors, the one or more instructions causing the one or
3 more processors to:
4 receive a command to generate a program fingerprint;
5 sort static data from within the program; and
6 create a program fingerprint using the sorted static data.

1 31. A computer-accessible medium according to Claim 30, wherein the static
2 data includes an import table.

1 32. A computer-accessible medium according to Claim 30, wherein the static
2 data includes function names corresponding to an import table from an executable of the
3 program.

1 33. A computer-accessible medium according to Claim 30, wherein the one or
2 more instructions that cause the one or more processors to sort static data further cause
3 the one or more processors to:

4 organize the function names from an import table corresponding to a program
5 executable in accordance with at least one predetermined criterion; and
6 link the organized function names.

1 34. A computer-accessible medium according to Claim 33, wherein the one or
2 more instructions that cause the one or more processors to create a program fingerprint
3 further cause the one or more processors to execute a hashing algorithm on the sorted
4 static data.

1 35. A computer-accessible medium having an application programming
2 interface (API), the API having one or more instructions to cause one or more processors
3 to:

4 receive a request to run a program;
5 generate a signature for the program;
6 compare the generated signature against a compilation of signatures corresponding
7 to restricted programs; and

enable only those programs for which the signature does not match with any of the
compiled signatures.

36. A computer-accessible medium according to Claim 35, wherein the one or
more instructions to generate a signature for the program cause the one or more
processors to sort a list of elements from an import table corresponding to an executable
of the program.

37. A computer-accessible medium according to Claim 36, wherein the one or
more instructions to generate a signature for the program cause the one or more
processors to hash a sorted list of function names from the import table.

38. A computer-executable medium according to Claim 35, wherein the API is
included in an operating system.

39. A computer-executable medium according to Claim 35, wherein the
operating system runs on a web server.

40. A computer-executable medium according to Claim 35, wherein the
operating system runs on an application server.

41. A license enforcement method, comprising:
generating a digital signature for each of a plurality of applications;
classifying each of the digital signatures in accordance with a licensing status for
the corresponding applications;

5 coding an operating system to:
6 include the classified digital signatures,
7 generate a digital signature for a requested application,
8 map the digital signature for the requested application to the classified
9 digital signatures, and
10 run the requested application when the digital signature for the requested
11 application does not map to digital signature classified as not being licensed.

1 42. A license enforcement method according to Claim 41, further comprising
2 downloading an updated list of the classified digital signatures to the operating system.